普通高等院校
计算机专业（本科）实用教程系列

数据结构实用教程

（Java语言描述）

徐孝凯　编著

清华大学出版社

内 容 简 介

本书是为全国高等院校计算机及相关专业开设数据结构课程而精心编著的一本实用教材。本书按照面向对象的程序设计方法，采用目前广泛使用的 Java 语言描述各种数据结构和运算方法，使得一种数据结构对应一种操作接口，进而通过不同的存储类型来实现。全书共分为 11 章，依次为绪论、集合、线性表、稀疏矩阵和广义表、栈和队列、树和二叉树、常用二叉树、图、图的应用、查找、排序。

全书内容丰富实用，结构层次分明，叙述简明易懂，运算方法分析透彻，所有算法描述都能够直接上机运行。这些显著特点都是作者多年来教材编写和教学经验的结晶，已经得到广大读者的认可。

本书可作为普通高等院校计算机及相关专业“数据结构”课程的教材或教学参考书。

为了配合使用本书，作者同时编写了相配套的《数据结构实用教程（Java 语言描述）习题参考解答》一书，一并出版。

图书在版编目（CIP）数据

数据结构实用教程：Java 语言描述 / 徐孝凯编著. --北京：清华大学出版社，2013.1
普通高等院校计算机专业（本科）实用教程系列
ISBN 978-7-302-30702-0

Ⅰ. ①数… Ⅱ. ①徐… Ⅲ. ①数据结构-高等学校-教材 ②JAVA 语言-程序设计-高等学校-教材
Ⅳ. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字（2012）第 278601 号

责任编辑：郑寅堃　王冰飞
封面设计：张　昱
责任校对：时翠兰
责任印制：沈　露

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
　　社 总 机：010-62770175　　**邮　　购**：010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
　　课 件 下 载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.5　　**字　　数**：522 千字
版　　次：2013 年 1 月第 1 版　　**印　　次**：2013 年 1 月第1 次印刷
印　　数：1～3000
定　　价：35.00 元

产品编号：047495-01

前　言

数据结构是普通院校计算机及相关专业的一门必修核心课程，主要讨论和研究从现实世界中抽象出来的数据的各种逻辑结构、在计算机中的存储结构，以及对其进行的各种处理运算的方法和算法。通过学习这门课程，使学生掌握如何利用计算机对数据进行有效的组织、存储和运算，为进一步学好后续各门计算机专业课程和进行软件开发打下良好的基础。

数据的逻辑结构大致分为集合结构、线性结构、树型结构和图型结构4种。数据在计算机中的存储结构大致分为顺序结构、链接结构、索引结构和散列结构4种。对数据进行的各种运算主要有插入运算、删除运算、查找运算、遍历运算、排序运算等。

开设数据结构课程需要借助一种计算机程序设计语言进行，在目前广泛使用的程序设计语言中，Java程序设计语言是最流行和面向对象程度最完善的语言。利用Java语言中的接口能够准确地描述任一种数据结构的逻辑定义和运算，利用任一种存储结构所定义的存储类型能够有效地实现接口。总之，采用面向对象的程序设计方法和使用Java语言讲授数据结构课程正在逐渐兴起。

本书采用面向对象的Java语言描述数据结构及其算法，是作者多年来教材编写和教学经验的结晶，是对以往利用计算机语言编写数据结构教材的丰富、发展和完善。全书共分为11章，分别为绪论、集合、线性表、稀疏矩阵和广义表、栈和队列、树和二叉树、常用二叉树、图、图的应用、查找、排序。在第1章中，结合数据表实例开门见山地给出了数据的4种逻辑结构，接着给出了算法的描述和评价等内容，为展开叙述后续章节的内容奠定基础。在第2章中，介绍了集合的抽象数据类型和接口，以及在顺序和链接存储结构下的具体实现。在第3章中，介绍了一般线性表的接口定义和在顺序与链接存储结构下的具体实现，以及有序线性表对一般线性表的继承与实现。在第4章中，介绍了稀疏矩阵的三元组表示和进行的各种矩阵运算，同时讨论了广义表的定义和运算。在第5章中，分别介绍了栈和队列的数据结构特点、运算方法和实现，着重讨论了栈与递归的关系、一些典型问题的递归求解方法、栈在算术表达式计算中的应用等内容。在第6章和第7章中，讨论了树和二叉树的定义、性质、存储结构和遍历方法，以及二叉搜索树、堆、哈夫曼树、平衡二叉树的定义、存储和运算。在第8章和第9章中，介绍了图的基本概念、图的3种存储结构、图的深度和广度优先遍历、图的最小生成树、最短路径、拓扑排序和关键路径等内容。在第10章中，主要介绍了二分查找、索引查找、散列查找、B树查找等内容。最后一章为排序，主要介绍了插入排序、选择排序、交换排序、归并排序和外存文件排序等内容。

书中的所有程序和算法都在Java开发和运行环境下调试通过，确保了算法的正确性和

有效性。为了配合本书教学，作者同时编写和出版了相配套的《数据结构实用教程（Java 语言描述）习题参考解答》一书，供同学们选用。在此书中，给出了每章内容的知识要点、课后练习题以及参考解答。练习题包括选择题、填空题、运算题、算法分析题、算法设计题等各种题型，通过做练习，能够巩固和提高对基本概念的理解深度和分析与设计算法的能力。

学习本教材应具备 Java 语言程序设计的基础知识，教学时数应安排在 80 学时左右，其中，讲授与上机实验的时数之比约为 2:1，有条件的学校要尽量多安排上机时间。

本书结构设计和内容编写全部由笔者完成，虽然本人有多年编写数据结构教材的经验，但由于水平有限，错误和不足之处在所难免，敬请专家和读者批评指正。

徐孝凯

2012 年 9 月

目　录

第 1 章　绪论 …… 1

1.1　基本概念 …… 1
1.2　算法描述 …… 11
1.3　算法评价 …… 13

第 2 章　集合 …… 20

2.1　集合的定义和运算 …… 20
2.1.1　集合的定义 …… 20
2.1.2　集合的抽象数据类型 …… 20
2.1.3　集合运算举例 …… 21
2.2　集合的顺序存储结构和操作实现 …… 23
2.3　集合的链接存储结构和操作实现 …… 30
2.3.1　链接存储的概念 …… 30
2.3.2　链接集合类的定义和实现 …… 33
2.4　集合应用举例 …… 39

第 3 章　线性表 …… 47

3.1　线性表的定义和运算 …… 47
3.1.1　线性表的定义 …… 47
3.1.2　线性表的抽象数据类型 …… 48
3.1.3　线性表运算举例 …… 49
3.2　线性表的顺序存储结构和操作实现 …… 52
3.3　有序线性表的定义和实现 …… 60
3.4　链接存储的一般概念和方法 …… 65
3.5　线性表的链接存储结构和操作实现 …… 70
3.6　有序线性表的链接存储结构和操作实现 …… 76
3.7　线性表应用举例——多项式计算 …… 78
3.7.1　多项式表示与求值 …… 78
3.7.2　两个多项式相加 …… 82

第 4 章　稀疏矩阵和广义表 …… 86

4.1　稀疏矩阵 …… 86

4.1.1 稀疏矩阵的定义……86
4.1.2 稀疏矩阵的转置运算……88
4.1.3 稀疏矩阵的加法运算……90
4.1.4 使用稀疏矩阵的程序举例……92
4.2 广义表……94
4.2.1 广义表的定义……94
4.2.2 广义表的存储结构……96
4.2.3 广义表类的定义……97
4.2.4 广义表的运算……99
4.2.5 简单程序举例……103

第 5 章 栈和队列……105

5.1 栈的定义和运算……105
5.2 栈的顺序存储结构和操作实现……106
5.3 栈的链接存储结构和操作实现……110
5.4 栈的简单应用举例……112
5.5 算术表达式的计算……116
5.6 栈与递归……124
5.7 队列……133
5.7.1 队列的定义和运算……133
5.7.2 队列的顺序存储结构和操作实现……134
5.7.3 队列的链接存储结构和操作实现……139

第 6 章 树和二叉树……141

6.1 树的概念……141
6.1.1 树的定义……141
6.1.2 树的表示……142
6.1.3 树的基本术语……142
6.1.4 树的性质……144
6.2 二叉树……145
6.2.1 二叉树的定义……145
6.2.2 二叉树的性质……145
6.2.3 二叉树的抽象数据类型……147
6.2.4 二叉树的存储结构……148
6.3 二叉树遍历……153
6.4 二叉树的其他运算……156
6.5 调试二叉树算法举例……160
6.6 树的存储结构和运算……161
6.6.1 树的抽象数据类型……161

6.6.2　树的存储结构……162
6.6.3　树的运算……166
6.6.4　调试普通树算法举例……171

第 7 章　常用二叉树……173

7.1　二叉搜索树……173
7.1.1　二叉搜索树的定义……173
7.1.2　二叉搜索树的抽象数据类型和链接存储类……174
7.1.3　二叉搜索树的运算方法……175
7.2　堆……181
7.2.1　堆的定义……181
7.2.2　堆的接口类……182
7.2.3　堆的存储结构和顺序存储类……182
7.2.4　堆的运算……184
7.3　哈夫曼树……189
7.3.1　基本术语……189
7.3.2　构造哈夫曼树……190
7.3.3　哈夫曼编码……193
7.4　平衡二叉树……195
7.4.1　平衡二叉树的定义……195
7.4.2　平衡二叉树的调整……197

第 8 章　图……202

8.1　图的概念……202
8.1.1　图的定义……202
8.1.2　图的基本术语……203
8.2　图的存储结构……205
8.2.1　邻接矩阵……205
8.2.2　邻接表……207
8.2.3　边集数组……208
8.3　图的抽象数据类型和接口类……209
8.4　图的邻接矩阵和邻接表存储类……210
8.5　图的遍历……214
8.5.1　深度优先搜索遍历……214
8.5.2　广度优先搜索遍历……217
8.5.3　非连通图的遍历……219
8.6　对图的其他运算的算法……219

第 9 章　图的应用……231

9.1　图的生成树和最小生成树……231

9.1.1 生成树的概念……231
9.1.2 普里姆算法……233
9.1.3 克鲁斯卡尔算法……237
9.2 最短路径……240
9.2.1 最短路径的概念……240
9.2.2 从一顶点到其余各顶点的最短路径……241
9.2.3 每对顶点之间的最短路径……246
9.3 拓扑排序……250
9.3.1 拓扑排序的概念……250
9.3.2 拓扑排序算法……252
9.4 关键路径……256

第 10 章 查找……264

10.1 查找的基本概念……264
10.2 顺序表查找……265
10.2.1 顺序查找……265
10.2.2 二分查找……267
10.3 索引查找……269
10.3.1 索引的概念……269
10.3.2 索引存储举例……270
10.3.3 索引查找算法……273
10.3.4 分块查找……274
10.4 散列查找……276
10.4.1 散列的概念……276
10.4.2 散列函数……278
10.4.3 处理冲突的方法……280
10.4.4 散列表的运算……284
10.5 B 树查找……293
10.5.1 B_树的定义……293
10.5.2 B_树查找……294
10.5.3 B_树的插入……296
10.5.4 B_树的删除……299
10.5.5 定义 B_树的类……302
10.5.6 B^+树简介……304

第 11 章 排序……306

11.1 排序的基本概念……306
11.2 插入排序……308
11.3 选择排序……309

11.3.1　直接选择排序……309
11.3.2　堆排序……310
11.4　交换排序……313
11.4.1　气泡排序……314
11.4.2　快速排序……315
11.5　归并排序……318
11.6　外排序……322
11.6.1　外排序的概念……322
11.6.2　外排序算法……323

参考文献……332

第1章 绪 论

1.1 基本概念

数据结构是计算机学科体系中一门最重要的核心课程，它主要研究如何能够把反映现实世界的抽象数据有效地组织存储到计算机系统中，并能够得到快速的访问和运算。

利用计算机存储数据，不仅要存储每个数据值本身，而且要存储数据之间相互联系而形成的逻辑关系，由此构成数据在计算机系统中的存储结构。数据的**存储结构**可以被概括为**顺序结构**、**链接结构**、**散列结构**和**索引结构**4种，它们之间的组合和嵌套可以形成更加复杂的数据存储结构。

对数据进行运算的方法简称**算法**，它是根据人们对数据进行处理的实际需要而逐渐产生、发展和丰富起来的。现在人们已经针对数据的不同处理需要探索出了各种相应和有效的算法，掌握这些现成的算法是进行各种软件开发和设计的基础和前提。要让计算机执行一种算法，必须把它由一般的文字语言或流程图描述转换成一种计算机语言的描述。所以在学习数据结构这门课程之前，必须学会一种计算机语言，这样才能够利用它作为工具编写出相应的程序设计算法，进而上机调试和运行程序（算法），实现人们利用计算机对数据进行快速、精确和自动处理的功能。

现在，可供学习的计算机语言很多，如 VB、C、C++、Delphi、Java 等。但目前最流行、应用最广泛、获取最方便的一种计算机语言是 Java 语言，并且它是一种完全体现最新程序设计思想的纯粹面向对象的程序设计语言。本书将采用 Java 程序设计语言来描述算法，使得数据结构课程的教学能够及时跟上学科发展和现实软件开发的需要。

在这门课程的教学中，经常需要使用到一些基本概念，下面对其作简要说明，为以后学习具体内容做准备。

1．数据

数据（data）是人们利用文字符号、数字符号以及其他规定的符号对现实世界的事物及其活动所做的抽象描述。例如，一个人的名字可以用一个字符串数据来描述，一条曲线可以用一个数组数据来描述，数组中的每一个元素用来存储曲线中对应点的坐标值和颜色编号。因此，一个文档、记录、数组、句子、单词、算式、符号等都统称为数据。在计算机科学领域，人们把能够被计算机加工的对象，或者说能够被计算机输入、存储、处理和输出的一切信息都叫做数据。

2．数据元素

数据元素（data element）简称**元素**，它是一个数据整体中可以标识和访问的数据个体。例如，对于一个文件数据来说，其中的每个记录就是它的数据元素；对于一个字符串数据

来说，每个字符就是它的数据元素；对于一个数组数据来说，每个下标位置上的表示值就是它的数据元素。数据和数据元素是相对而言的。例如，对于一个记录数据来说，它是所在文件的一个数据元素，而它相对于所含的每个数据项而言又是整体数据，数据项是记录数据的一个成分（域、元素）。因此，在本书中，对数据和数据元素这两个术语的使用不加以严格区分，读者应根据上下文含义进行理解。

3．数据记录

数据记录（data record）简称**记录**，它是数据处理领域组织数据的基本单位，数据中的每个数据元素（个体）在许多应用场合被组织成记录的结构。一个数据记录由一个或多个**数据项**（item）所组成，每个数据项可以是简单数据项（即不可再分，如一个数值、一个字符等），也可以是组合数据项（即数组或记录）。就拿一张图书目录表来说，每个数据记录表示一本图书的有关信息，如表 1-1 所示。

表 1-1　图书目录表

登录号	书　　号	书　　名	作者	出版社	定价
00001	ISBN 978-7-302-18161-3	Java 语言程序设计	陈明	清华大学	26.00
00002	ISBN 978-7-302-15579-9	Java 程序设计教程（第 2 版）	雍俊海	清华大学	49.00
00003	ISBN 978-7-302-15761-8	C++语言基础教程（第 2 版）	徐孝凯	清华大学	32.00
00004	ISBN 978-7-304-04066-6	C 语言程序设计	任爱华	中央电大	21.00
00005	ISBN 7-304-02494-1	数据库基础与应用	刘世峰	中央电大	28.00
00006	ISBN 7-04-014601-0	大学计算机基础教程	王移芝	高等教育	27.50
00007	ISBN 7-04-007494-X	数据库系统概论	王　珊	高等教育	25.10
…	…	…	…	…	…

在表 1-1 中，第一行为表目行，又称目录行，它给出了该表中每条记录的组织结构。从表目行向下的每一行为一条包含具体值的记录，它给出了相应的一本图书的信息；每一列为一个数据项，它描述了图书中的一种属性。每条记录由 6 个数据项所组成，其名称分别为登录号、书号、书名、作者、出版社和定价，前 5 个数据项均为字符串，后一个数据项为数值。

在一张表中，若所有记录的某个数据项（属性）的值均不同，也就是说，每个值能够唯一地标识一个记录，则称这个数据项为记录的关键数据项，简称**关键项**（key item），关键项中的每个值被称为所在记录的**关键字**（key）。在表 1-1 中，登录号数据项的值均不同，所以可把登录号作为记录的关键项，其中的每一个值就是所在记录的关键字。例如，00002 为第 2 条记录的关键字，00005 为第 5 条记录的关键字。表 1-1 中的书号数据项的值也各不相同，所以每个记录的书号也是关键字，根据每个具体的书号也能够唯一地标识一本图书。

对于一个通常使用的表格数据（文件），能作为关键项的数据项可能没有，可能只有一个，也可能多于一个。当没有时，可把多个有关的数据项联合起来，构成一个组合关键项，用组合关键项中的每一个组合值来唯一地标识一个记录，该组合值就是所在记录的关键字。

引入了记录的关键项和关键字后，为以后叙述简便起见，经常利用一个关键项中的所

有关键字的集合来代替整个数据表，利用一个关键字来代替所在的记录，而把记录中的其他数据项忽略掉。

4．数据结构

数据结构（data structure）是指数据及其相互之间的联系（逻辑关系）。数据是对现实世界中的事物及其活动的抽象描述，而任何事物及其活动都不是孤立存在的，都是在一定意义上相互联系、相互影响的，所以数据之间必然存在着客观或主观上的联系。数据之间的相互联系被称为数据的**逻辑结构**。在计算机中存储数据时，不仅要存储数据的值，而且要存储数据之间的联系（逻辑结构）。数据的逻辑结构被存储到计算机的存储器中，就形成了数据的存储结构。数据的逻辑结构被概括为**集合结构**、**线性结构**、**树型结构**、**图型结构** 4 种，由它们的组合和嵌套可以构成更为复杂的逻辑结构。通常所说的数据结构是指逻辑结构，但有时也包括存储结构在内，读者应结合上下文理解其含义。

为了更确切地描述数据的逻辑结构，通常采用二元组表示：

B=(K, R)

B 是一种数据结构（逻辑结构），它由数据元素的集合 K 和 K 上二元关系的集合 R 所组成。其中：

$K=\{k_i \mid 1\leqslant i\leqslant n, n\geqslant 0\}$

$R=\{R_j \mid 1\leqslant j\leqslant m, m\geqslant 0\}$

其中，k_i 表示集合 K 中的第 i 个数据元素，n 为 K 中数据元素的个数，特别地，若 n=0，则 K 是一个空集，此时 B 也就无结构而言，也可以认为它具有任一结构；R_j 表示集合 R 中的第 j 个二元关系（可直接简称为关系），m 为 R 中关系（二元关系）的个数，特别地，若 m=0，则 R 是一个空集，表明不考虑集合 K 中的元素之间存在着任何关系，元素之间是彼此独立的，把这种 R 为空集的数据结构称为集合结构。

在本书所讨论的各种数据结构中，一般只讨论 m=1 的情况，即 R 中只包含有一个关系，此时 $R=\{R_1\}$，并且为了简化起见，直接把这个关系 R_1 用标识符 R 来表示。对于包含有多个关系的数据结构，可以首先对每一个关系的情况分别进行讨论，然后再整体讨论。

K 上的一个二元关系 R 是**序偶**的集合。对于 R 中的任一序偶$<x,y>(x,y\in K)$，把 x 叫做序偶的第一元素，把 y 叫做序偶的第二元素，又称序偶的第一元素为第二元素的**前驱**，更准确地说是直接前驱，称第二元素为第一元素的**后继**，更准确地说是直接后继。例如，在$<x,y>$的序偶中，x 为 y 的前驱，而 y 为 x 的后继。

一种数据结构还能够利用图形直观地表示出来，图形（示意图）中的每个结点（顶点）对应着一个数据元素，两结点之间带箭头的连线（又称做**有向边**或**弧**）对应着关系中的一个序偶，其中，序偶的第一元素为有向边的起始结点，第二元素为有向边的终止结点，即箭头所指向的结点。

作为例子，下面根据表 1-2 来构造出一些典型的数据结构。

表 1-2　教务处人事简表

职工号	姓　名	性别	出生日期	职务	部门
01	万明华	男	1965.03.20	处长	教务处
02	赵　宁	男	1973.06.14	科长	教材科

续表

职工号	姓　名	性别	出生日期	职务	部门
03	张　利	女	1969.12.07	科长	考务科
04	赵书芳	女	1977.08.05	主任	办公室
05	刘永年	男	1964.08.15	科员	教材科
06	王明理	女	1980.04.01	科员	教材科
07	王　敏	女	1977.06.28	科员	考务科
08	张　才	男	1972.03.17	科员	考务科
09	马立仁	男	1980.10.12	科员	考务科
10	邢怀常	男	1981.07.05	科员	办公室

表 1-2 中共有 10 条记录，每条记录都由 6 个数据项所组成，由于每条记录的职工号各不相同，所以可把每条记录的职工号作为该记录的关键字，并在下面的例子中，将用记录的关键字来代表所在记录。

例 1-1　一种数据结构 set=(*K*,*R*)，其中：

K={01,02,03,04,05,06,07,08,09,10}

R={}

在数据结构 set 中，只存在元素的集合，不存在关系的集合，或者说关系的集合为空，这表明我们只考虑表 1-2 中的每条记录，并不考虑它们之间的任何关系。称具有这种特点的数据结构为**集合结构**。对于集合结构，元素之间的位置无关紧要，人们对它按任何次序排列都是允许的，可以按照关键字的升序排列，也可以按照关键字的降序排列，还可以按照其他任意次序排列，这应根据处理问题的需要选择决定。

例 1-2　一种数据结构 linearity=(*K*,*R*)，其中：

K={01,02,03,04,05,06,07,08,09,10}

R={<05,01>,<01,03>,<03,08>,<08,02>,<02,07>,<07,04>,
<04,06>,<06,09>,<09,10>}

对应的图形如图 1-1 所示。

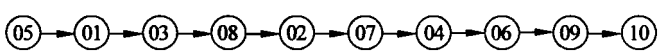

图 1-1　数据的线性结构示意图

结合表 1-2，读者不难看出 *R* 关系是按照职工年龄从大到小排列的线性关系。

在 linearity 数据结构中，每个数据元素有且仅有一个直接前驱元素（除结构中第一个元素 05 外），有且仅有一个直接后继元素（除结构中最后一个元素 10 外）。这种数据结构的特点是数据元素之间的 1 对 1（1:1）联系，称这种联系为**线性关系**，人们把具有这种特点的数据结构称为**线性结构**。所以，例 1-2 所示的数据结构为一种线性结构。

例 1-3　一种数据结构 tree=(*K*,*R*)，其中：

K={01,02,03,04,05,06,07,08,09,10}

R={<01,02>,<01,03>,<01,04>,<02,05>,<02,06>,<03,07>,
<03,08>,<03,09>,<04,10>}

对应的示意图如图 1-2 所示。

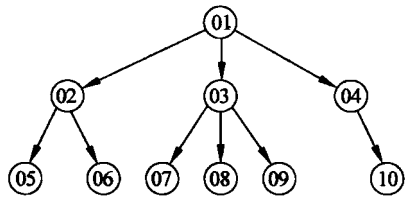

图 1-2　数据的树型结构示意图

结合表 1-2，读者不难看出 R 关系是单位员工之间领导与被领导的关系。

图 1-2 像倒着画的一棵树，在这棵树中，最上面一层的一个没有前驱只有后继的结点叫做**树根结点**，最下面一层的只有前驱没有后继的结点叫做**树叶结点**，除树叶结点之外的所有结点都叫做**树枝结点**，树根结点是一个特殊的树枝结点。在一棵树中，每个结点有且只有一个前驱结点（除树根结点外），但可以有任意多个后继结点（树叶结点可看做具有 0 个后继结点）。这种数据结构的特点是数据元素之间的 1 对 N（1:N）的联系（N≥0），即一对多的联系。人们把这种数据联系称为**层次关系**，把具有这种特点的数据结构叫做**树型结构**，简称**树结构**或**树**。所以，例 1-3 所示的数据结构为一种树结构。

例 1-4　一种数据结构 graph=(K,R)，其中：

K={01,02,03,04,05,06,07}

R={<01,02>,<02,01>,<01,04>,<04,01>,<02,03>,<03,02>,<02,06>,<06,02>,
<02,07>,<07,02>,<03,07>,<07,03>,<04,06>,<06,04>,<05,07>,<07,05>}

对应的图形如图 1-3 所示。

从图 1-3 可以看出，R 是 K 上的对称关系，即对于 K 上的任何一个序偶<x,y>，其中必然存在着另一个序偶<y,x>。为了简化起见，把<x,y>和<y,x>这两个对称序偶用一个无序对(x,y)或(y,x)来代替；在示意图中，把 x 结点和 y 结点之间两条相反的有向边用一条无向边来代替。这样，R 关系可改写为：

R={(01,02),(01,04),(02,03),(02,06),(02,07),(03,07),(04,06),(05,07)}

对应的图形如图 1-4 所示。

如果说在这个 R 中的每个序偶里的两个元素所代表的人员是好友的话，那么 R 关系就是员工之间目前已存在的好友关系。

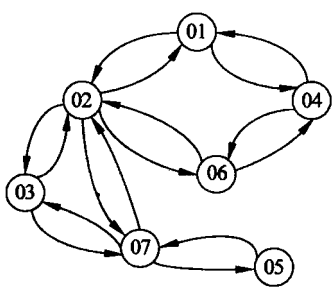

图 1-3　数据的图型结构示意图

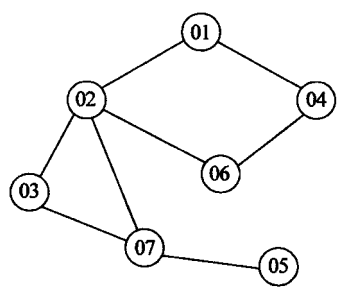

图 1-4　图 1-3 的等价表示

从图 1-3 或图 1-4 可以看出，结点之间的联系是 M 对 N（M:N）的联系（M≥0，N≥0），

即多对多的联系，我们把这种联系称为**网状关系**，也就是说，每个结点可以有任意多个前驱结点和任意多个后继结点。人们把具有网状关系特点的数据结构叫做**图型结构**，简称**图结构**或**图**。所以，例 1-4 所示的数据结构为一种图结构。

由图型结构、树型结构和线性结构的定义可知，树型结构（一对多联系）是图型结构（多对多联系）的特殊情况，而线性结构（一对一联系）又是树型结构的特殊情况，当然更是图型结构的特殊情况。为了区别于一对一数据联系的线性结构，把树型结构和图型结构统称为**非线性结构**。线性结构和非线性结构相对于集合来说，其元素之间存在着固定联系，而集合结构中的元素之间无联系，或者说无固定联系。

例 1-5 一种数据结构 B=(*K*,*R*)，其中：

$K=\{k_1,k_2,k_3,k_4,k_5,k_6\}$

$R=\{R_1,R_2\}$

$R_1=\{<k_3,k_2>,<k_3,k_5>,<k_2,k_1>,<k_5,k_4>,<k_5,k_6>\}$

$R_2=\{<k_1,k_2>,<k_2,k_3>,<k_3,k_4>,<k_4,k_5>,<k_5,k_6>\}$

在图形表示中，若用实线表示关系 R_1 中的序偶，用虚线表示关系 R_2 中的序偶，则对应的图形如图 1-5 所示。

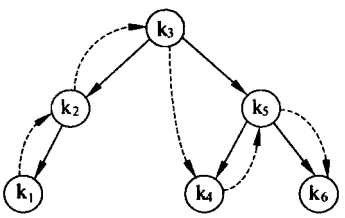

图 1-5 带有两个关系的数据结构示意图

从图 1-5 可以看出，数据结构 B 是一种非线性的图型结构。但是，若只考虑关系 R_1 则为树型结构，若只考虑关系 R_2 则为线性结构。

若一个数据整体中包含多个关系，人们通常把它分解为多个数据结构来看待和处理。例如，对于上面表 1-2，根据处理不同问题的需要，就可能同时存在着按年龄排列建立的线性关系、按领导和被领导建立的层次关系、按好友联系建立的网状关系、不考虑任何联系的无关系等。使它们每个关系分别同数据集合构成二元组，从而产生相应不同的数据结构，然后再进行相应的数据处理和运算。

5. 数据类型

数据类型（data type）是对数据的取值范围、每个数据的结构以及允许施加运算的一种定义。每一种计算机高级语言都定义有自己的数据类型。在通用的计算机高级语言中，一般都具有整数、实数（浮点数）、枚举、字符、字符串、指针、数组、记录（结构）、文件等数据类型。例如，整数类型中的每个值（整数）在计算机系统中通常用两个或 4 个字节存储（表示），若采用两个字节，则整数类型的表示范围为-2^{15}～$2^{15}-1$，即–32 768～32 767；若采用 4 个字节，则整数类型的表示范围为-2^{31}～$2^{31}-1$，即–2 147 483 648～2 147 483 647。对整数类型的数据允许施加的运算通常有单目取正或取负运算，双目加、减、乘、除、取模等运算，双目等于、不等于、大于、大于等于、小于、小于等于等关系（比较）运算，

以及赋值运算等。字符类型中的每个值（字符）在计算机存储系统中通常用一个或两个字节表示，若采用一个字节，则字符类型的表示范围为 0～255 或–128～127，能够至多对 256 种不同的字符进行编码。对字符类型的数据允许进行的运算主要为赋值和各种关系运算。字符串类型中的每个值（字符串）为字符的顺序排列结构（即线性结构），字符串类型的取值范围很广泛，任何一个字符序列，如姓名、地址、编号等非数值的数据信息都是一个字符串。对字符串的运算主要有求字符串的长度、字符串的复制、字符串的连接、字符串的比较等。

1）数据类型分类

数据类型可以被分为**简单类型**和**结构类型**两大类。简单类型中的每个值通常只作为整体使用，如一个整数、实数、字符、指针、枚举量等都是相应简单类型中的数据。结构类型由简单类型按照一定的规则构造而成，并且结构类型仍可以包含结构类型，所以一种结构类型中的数据（即结构数据）可以分解为若干个简单数据或结构数据，每个结构数据仍可再分。例如，数组就是一种具有线性数据结构的结构数据类型，数组中的每一种数组值包含有固定个数的同一类型的数据，每个数据（元素）都可以通过下标运算符直接访问。记录也是一种具有线性数据结构的结构数据类型，记录中的每一个记录值包含固定个数的不同类型的数据成员，每个数据成员都可以通过成员运算符和成员名直接访问。另外，像字符串和文件也都是计算机高级语言中经常提供的结构数据类型。

2）数据类型与数据值

无论是简单类型还是结构类型都有“**型**”和“**值**”的区别，“型”是“值”的抽象定义，一种数据类型中的任一数据称为该类型中的一个值，又被称为**实例**，该值（实例）与所属数据类型具有完全相同的逻辑结构，数据类型所规定的运算都是在值上进行的。所以在一般的叙述中，并不明确指出是数据“型”还是数据“值”，读者应根据上下文加以理解。例如，提到记录时，当讨论的是记录结构则认为是记录型，当讨论的是具体一条记录时则认为是记录值。

3）数组的逻辑结构

对于在计算机语言中使用的数组、记录、字符串和文件等结构数据类型，每个值中的元素（成分）是按位置前后有序排列的，所以它们各自都具有线性数据结构。数组的逻辑结构是线性数据结构，采用二元组形式可描述为：

array=(A,R)，其中：

A={a[i] | 0≤i≤n–1,n≥1}

R={<a[i],a[i+1]> | 0≤i≤n–2}

a[i]为数组中的下标为 i 的元素，n 为大于等于 1 的整数，用来表明数组中元素的个数，即数组长度，数组元素的下标从 0 到 n–1，数组中前后相邻位置上的两个元素为一个序偶，其前一元素 a[i]是后一元素 a[i+1]的前驱，而 a[i+1]是 a[i]的后继，第一个元素 a[0]无前驱元素，最后一个元素 a[n–1]无后继元素。

按数组下标的个数，可把数组分为一维、二维、三维等。一维数组中的每个元素只包含一个下标，二维数组中的每个元素包含两个下标，第一个称为行下标，第二个称为列下标。

二维数组被看作是一维数组的推广或嵌套，即首先把它看作是按行下标有序的线性结

构，即只含有行下标的一维数组；对于其中的每个元素又都是按列下标有序的线性结构，即只含有列下标的一维数组。也就是说，二维数组具有双重的线性结构。例如，对于一个二维数组 b[*m*][*n*]，可首先看作为一维数组 b[*m*]，所含元素依次为 b[0]、b[1]、…、b[*m*–1]，其中每一个元素 b[*i*]（0≤*i*≤*m*–1）又都是一个含有 *n* 个元素的一维数组，所含元素依次为 b[*i*][0]、b[*i*][1]、…、b[*i*][*n*–1]。

同样，三维数组包含 3 个下标，每个元素的位置由一组 3 个下标值唯一确定。三维数组是一维数组的三层嵌套结构。例如，对于一个三维数组 c[*p*][*m*][*n*]，首先被看作为一维数组 c[*p*]，所含元素依次为 c[0]、c[1]、…、c[*p*–1]，其中每一个元素 c[*k*]（0≤*k*≤*p*–1）又都是一个含有 *m* 个元素的一维数组，所含元素依次为 c[*k*][0]、c[*k*][1]、…、c[*k*][*m*–1]，这里的每一个元素 c[*k*][*i*]（0≤*i*≤*m*–1）又都是一个含有 *n* 个元素的一维数组，所含元素依次为 c[*k*][*i*][0]、c[*k*][*i*][1]、…、c[*k*][*i*][*n*–1]。

4）数组的存储结构

数组是在各种计算机语言中都被定义和使用的一种结构数据类型，当然它既包含逻辑结构，也包含存储结构以及相应的运算。数组的存储结构是**顺序结构**，即数组中第 *i*+1 个元素紧接着存储在第 *i* 个元素的存储空间位置的后面。例如，对于一维数组 a[*n*]，则每个元素 a[*i*]的存储位置的首字节地址为：

Address(a[*i*])=Loc(a)+*i*×*L*　　（0≤*i*≤*n*–1）

其中，Loc(a)表示数组 a 的存储空间的首地址，*L* 表示数组 a 中元素类型的大小，即每个元素所占用的存储空间的字节数。由上述公式可知：元素 a[0]的存储地址为 Loc(a)，它就是整个数组的开始存储地址，a[1]的存储地址为 Loc(a)+1×*L*，a[2]的存储地址为 Loc(a)+2×*L*，最后一个元素 a[*n*–1]的存储地址为 Loc(a)+(*n*–1)×L。

再来分析一下二维数组的存储结构。假定一个二维数组为 b[*m*][*n*]，每一行元素 b[*i*]的存储位置（即存储该行 *n* 个元素的首字节地址）为：

Address(b[*i*])=Loc(b)+*i*×RS　　(0≤*i*≤*m*–1)

其中，Loc(b)表示二维数组 b 的存储空间的首地址，RS 表示顺序存储一行 *n* 个元素所占用存储空间的字节数，它等于每个元素所占用的字节数 *L* 与一行上元素的个数 *n* 的乘积。因此上述计算公式可改写为：

Address(b[*i*])=Loc(b)+*i*×*n*×*L*　　(0≤*i*≤*m*–1)

对于二维数组 b 中的下标为 *i* 的行，其中下标为 *j* 的元素 b[*i*][*j*]的存储位置为：

Address(b[*i*][*j*])=Loc(b)+*i*×*n*×*L*+*j*×*L*　　(0≤*i*≤*m*–1,0≤*j*≤*n*–1)

可见，二维数组的存储结构是嵌套的顺序存储结构。

对于三维或更高维的数组，读者可进行类似的逻辑结构和存储结构的分析。

5）数组的运算

对数组的运算有许多，最常用的就是访问数组元素和返回数组长度。在所有的计算机语言中，都是通过数组名、下标和下标运算符（中括号）来访问数组元素的。

6．抽象数据类型

抽象数据类型（Abstract Data Type，ADT）由一组带有结构的数据和在该组数据上的操作集所组成。抽象数据类型包含一般数据类型的概念，但含义比一般数据类型更广、更